Analysing Web Traffic

Richard Foster

BSc Computing with Management Studies

2004/2005

Summary

This project aimed to produce an application that would assist website maintainers and developers with the task of analysing the information that is stored in the log files that websites produce. The requirements for the application were identified after reviewing the existing applications that analysed web traffic and highlighting any features that they failed to support.

The application that has been produced operates from the command line on either Windows or Linux operating systems and it provides the user with the ability to quickly and simply analyse the contents of the error log file. This is achieved by providing various summaries and reports for the user to view, the ability to search for errors by entering various parameters and a range of settings enables the user to tailor the application to meet their needs.

Acknowledgements

I would like to thank the Tony Jenkins for all of the guidance, advice and support that he has given over the course of this project.

I would also like to thank Bill Whyte for his feedback, constructive comments and for assessing this project.

Thanks must also go to Toby Chantrell, Tara Frost, Matt Manley and Will Jackson for their continuous support, motivation and help. Also thanks to them for generally putting up with me throughout the year.

Thanks must also go to those who helped with the evaluation of the finished application, and also everyone else who helped me.

Finally I would like to thank my family for everything.

Cheers.

Contents

Chapter 1

Introduction

1.1 Statement of the problem

The aim of this project was to first of all look at the problems concerning the analysis of log files that are produced by website servers. Once this had been done existing applications, that are available so that people can use to analyse log files, were reviewed in order to establish whether or not there were any weaknesses. It was then possible to develop a simple application that solved some of these weaknesses and improved upon the existing applications.

The main target user group for the application are people who maintain their own website, or their organisations website and wish to analyse the data that is generated when people visit the website.

The application that has been developed is geared toward quick and easy analysis of website statistics, and that can be used by a person with limited technical knowledge.

1.2 Background to the Problem

1.2.1 World Wide Web Statistics

Since the inception of the World Wide Web in 1993 the number of users along with the number of websites and web pages has grown exponentially. In March 2005 [1] estimates that out of the world's 6.4billion population, around 0.9billion people use the Internet, and they state that over the last five years alone (between 2000 and 2005) Internet usage has grown by one hundred and forty six percent. In terms of the number of websites and pages [2] estimates that in January 2005 there were 317,646,084 hosts registered on the Domain Name System (DNS), an increase of thirty six percent on the figure for the previous year and this is an extremely high amount when compared to the initial 1,313,000 hosts in 1993 (see figure 1.2.1).

There are many reasons for the growth in Internet usage, ranging from the growth in the number of homes that now have computers, the falling price of technology and way that technology is now being embraced in educational institutes across the world.

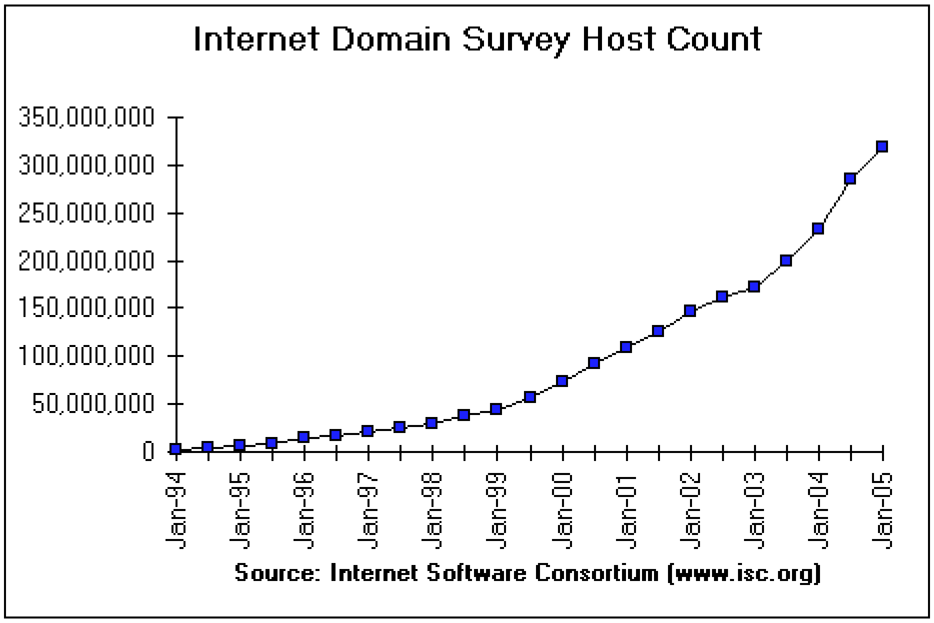


Figure 1.1 Growth in Internet Hosts from 1994 to 2005

Another reason for this growth is the advent of e-business with more and more companies using the Internet in order to exploit the global marketplace and negotiate the best possible deals at a fraction of what it would cost if someone had to travel to the other side of the world to make the same deal.

All of the servers that host these websites have the ability to produced log files that record various statistics relating to the visitors to the sites, and the fact that there is a substantial number of both websites and web users then this means that there must be many visits made by users to websites and so therefore there will be a lot of information stored about these visits.

1.2.2 The Importance of Log files

There are many reasons why log files are important to the owners of websites; this is because they contain a wealth of information about the people who use the site. Perhaps the most important reason of all is to see exactly who is using your website and find out more about them. Information that can be identified about the users includes what browsers and operating systems they use, where they are located, etc.

With this information the website can then be tailored towards the most popular browser and operating systems, however it should be remembered that not everyone uses the same browser or operating system, therefore its important not to alienate people by forcing them to view on Internet Explorer, for example. The location of the visitors to the site is important because if a website was in French and it was viewed by a large percentage of people from say America, then it might be worth creating an English version of the site as well. Another factor on location is that if people are only viewing the site from a concentrated geographical region (just one county for example) then it may be worth while advertising the website in other regions in attempt to attract new users. However this depends on the website and what its purpose is.

Another important factor of log files is that they identify the most popular features of a website (i.e. the pages that are visited more frequently), this means that the owners of the site know which areas of the site are popular, and therefore this means that resources can be targeted towards these areas, however it also identifies the less popular areas and these can either be improved or removed from the site. This is important because a website needs to be regularly updated and improved in order to attract repeat visitors.

Overall the main reason why log files are important is that they allow webmasters to tailor their websites so that they meet the needs of the majority of the users and this will hopefully result in continued usage by these people and also by identifying weaker areas of the website this will allow the webmaster to improve on these and attract new users. Also marketing is an important aspect, particularly in e-commerce websites, because the more you know about people then the easier it will be to identify and satisfy their needs.

1.3 Objectives

This Project was broken down into the following objectives that all needed to be achieved for successful completion:

- Develop an understanding of existing web traffic analysis applications.
- Identify any weaknesses that exist with existing web traffic applications.
- Identify the requirements and specification for an application that would improve on one or more than one of these weaknesses.
- Design the interface and program structure for the application.
- Implement an application that will meet the requirements and specification.
- Evaluate how the application performs in real world situations.

1.4 Project Requirements

From the project objectives a set of minimum project requirements and possible future enhancements were developed.

The minimum requirements were:

1. Review of existing web traffic analysis tools to establish their features.
2. Evaluate findings of 1 to identify any weaknesses with these tools.
3. Design and implement a basic application, which will solve one of the weaknesses identified in 2.
4. Evaluation of the application and suggest any improvements.

The possible enhancements were;

1. Extend the application to analyse mail server and FTP log files.
2. Extend the application to allow the users to customise the data output and layout, for example different styles of tables for results, etc.
3. Extend the application to provide the ability to compare different log files and identify similar trends.
4. Enable the application to function on any computer that is connected to the Internet.

1.5 Project Schedule

To ensure that I allocated enough time to each part of the project a schedule based on the five stages of the development life cycle was established at the beginning of the project. This was also used to set deadlines for the completion of the different sections and also allowed me to successfully plan work for other modules around the project. The Gannt chart for the schedule can be found in Appendix B.

The project was divided into two sections due to the Christmas holidays and examination revision period. The first semester was dedicated to background research and analysis of the problem including development of a set of product requirements. This meant that the second semester would focus on the design and implementation of the actual system.

The schedule was flexible to provide time to reflect on the recommendations made by the project assessor after submission of the mid-project report and there was also time allowed for the project to recover from any unexpected delays. The January exam period was earmarked for revision only and therefore no project work needed to be done until the exams were over. The second semester schedule was structured to leave a large period of time to write up the project.

1.6 Deliverables

The deliverables produced in the completion of this project are:

- Final Project Report – documentation for the project.
- Software application – that solves the problems identified.
- User manual –A manual to help people use the application.

1.7 Evaluation Criteria

In order to perform a successful evaluation of both the product and the project, the following criteria have been identified. In terms of the product, the application that is created should improve on existing solutions and therefore offer the project owners, webmasters and website maintainers, a way of analysing web traffic that is a) not available with any other application and b) will be simple and quick to use. The application will be tested in a real world environment by different evaluators. The project itself will be evaluated in terms of whether the objectives and the minimum requirements identified in this chapter have been achieved, and whether this has happened on time and before the deadlines set.

1.8 Milestones and Personal Goals

1.8.1 Milestones

The following milestones were identified to provide specific dates by which certain tasks had to be completed. This is because without deadlines it is difficult to self motivate, and so the inclusion of these should result in a well developed project and reduce the chance of falling behind:

- 09/12/2004 – Submit Mid-project Report paper copy and electronic copy.
- 24/01/2005 – Collect report feedback.
- 11/03/2005 – Submit Table of contents and Draft Chapter.
- 18/03/2005 – Completion of progress meeting.
- 27/04/2005 – Submit project report plus photocopy.
- 29/04/2005 – Submit report pdf electronically.
- 27/05/2005 – Receive feedback on project.

1.8.2 Personal goals:

- To finish all background research and have started the analysis stage by the end of December.
- To finish all of the coding by the middle of March (Easter).
- Write up a draft project report for the end of the Easter break so that it can be checked and corrected.

Chapter 2

Background Reading

2.1 What is a log file?

In order to understand the problems of analysing web traffic, the issue of what log files are must first be understood.

When a file is requested from a website, the web server will store details of these requests and keep a record of them in one file. These files are called log files and they are usually stored with a .log extension. However details of which pages were requested and when, are not the only details that are stored. The log files also contain various other pieces of information about the people or programs that requested them.

The Leeds CAMRA website [3] produces two types of log files, the Access log file and the Error log file. These store different types of information regarding when the site is accessed and when errors occur. The next step is to look at each of the log files and identify exactly what details they store.

2.1.1 access.log

This log file stores all of the details concerning the times when the web site has been accessed. Figure 1 gives an example of a log file, and it can be seen that the access log file looks rather complicated with very few details being apparent, and with over one hundred thousand entries for the last fifteen months there is quite a lot of information to be uncovered.

However thankfully each of these entries has a common format and while nothing is immediately apparent from the data if we look more closely at the highlighted entry from this log file the information in it becomes a bit clearer.

```
195.93.34.9 - - [22/Sep/2004:08:31:06 +0100] "GET
/Download/Wallpaper/Thumb/chequersThumb.jpg HTTP/1.0" 200 43512 "http://www.leeds-
camra.com/Download/" "Mozilla/4.0 (compatible; MSIE 5.5; AOL 8.0; Windows 98; Win 9x
4.90; FunWebProducts)" 0 www.leeds-camra.com
```

From just glancing at this entry it is easy to see when the user accessed the file, what browser they were apparently using (Mozilla 4.0) and also the file that they viewed (an image called chequersThumb.jpg). However what the rest means is not really straightforward. After looking at various articles [4, 5, 6] relating to the format of log it became apparent exactly what the each entry was actually telling me.

The following table defines each part of the log file and explains what it means.

Log File Entry:

```
195.93.34.9 - - [22/Sep/2004:08:31:06 +0100] "GET
/Download/Wallpaper/Thumb/chequersThumb.jpg HTTP/1.0" 200 43512 "http://www.leeds-
camra.com/Download/" "Mozilla/4.0 (compatible; MSIE 5.5; AOL 8.0; Windows 98; Win 9x
4.90; FunWebProducts)" 0 www.leeds-camra.com
```

Access Log Field	Example	Explanation
Hostname or IP address of client	195.93.34.9	This identifies the person (user) or machine that made the request
RFC91 Information	-	Remote log name of the user
Username	-	The username as which the user has authenticated himself
Date/Time	[22/Sep/2004:08:31:06 +0100]	This gives the date and time (dd/mmm/yyyy:hh:mm:ss) when the request was made
Request followed by Protocol	"GET /Download/Wallpaper/Thumb/chequersThumb.jpg HTTP/1.0"	The file that was requested and the HTTP protocol which was used to make the request
Status Code	200	The HTTP status code returned back to the user. Definitions are available from RFC2616 [7]
Bytes transferred	43512	The Number of bytes transferred during the request
Referrer	"http://www.leeds-camra.com/Download/"	The page which links to/includes the object that was requested
Browser	"Mozilla/4.0 (compatible; MSIE 5.5; AOL 8.0; Windows 98; Win 9x 4.90; FunWebProducts)"	The tag that the browser uses to identify itself, the browser does not have to tell the truth.
Unknown	0	This is an unknown variable but in this log file it is always 0.
Virtual Host	www.leeds-camra.com	The virtual host where the website is located

2.1.2 error.log

This file not surprisingly stores the information regarding all of the requests, which lead to errors occurring. To understand what the different fields meant it was useful to refer to the Apache Log Files documentation [5]. This error log file is fairly straightforward as there are not that many different log fields as you can see from the following entry:

```
[Sun Oct 24 12:40:35 2004] [error] [client 81.152.5.154] File does not exist:
/home/tony/www/leeds-camra.com/www/img/daveCider.jpg
```

From this line of code it is clear that there are only five different fields and all of them are fairly self-explanatory.

Error Log Field	**Example**	**Explanation**
Date/Time	[Sun Oct 24 12:40:35 2004]	This gives the date and time (day mon dd hh:mm:ss yyyy) of the error
Log Level Directive	[error]	Lists the severity of the error using the LogLevel directive [8]
Client	[client 81.152.5.154]	This identifies the client who made the request
Error message	File does not exist:	States what exactly the error is. In this case a particular file has not been found
Requested document	/home/tony/www/leeds-camra.com/www/img/daveCider.jpg	The file-system path of the requested document.

2.2 Systems Development Methodologies

The aim of the project is to create an application to analyse the log files that are generated by the Leeds CAMRA website. Therefore in order to create an application the author first needed to research into the various different systems development methodologies that are used in the creation of any application. To obtain an understanding of the different methodologies [9] was referred to. There are many different types of software development models, most of those that exist today developed from the 3 following approaches:

- Ad-Hoc Development
- The Waterfall Model (Systems Life Cycle)
- Iterative Development

These three models will now be investigated to establish which would be the most suitable for the project.

2.2.1 Ad-Hoc Development

This method was one of the earliest systems development models and it tended to occur in an unorganised and somewhat chaotic way. The dictionary defines ad hoc as being, “For the specific purpose, case, or situation at hand and for no other.” Therefore this points to the fact that tasks carried out are concerned with solving the problem and nothing else, as a result not much planning takes place as people tend to just try things out on the spur of the moment. Therefore as every project is different then there are no ground rules for this type of method.

In conclusion it would appear that this method is not really suitable for this type of project. This is because there needs to be a clear and logical plan so that the author can meet all of the deadlines that have been formally set.

2.2.2 The Waterfall Model

This form of methodology is the earliest method of structured system development, and it is still in use today to solve many problems. In this model the creation of a system to solve a problem is broken down into the following stages:

- **System Conceptualisation:** This involves breaking down the problem to determine exactly what the application needs to do in order to solve the problem.
- **System Analysis:** This involves the gathering of the system requirements and looking at how they will be accommodated in the system.
- **System Design:** This step will detail how the various components of the system will be constructed in order to perform the specific tasks. This also looks at how the system will be used and how it will ‘feel’.
- **Coding:** This is the stage of the model where the system software is created.
- **Testing:** This is where the software that has been created is tested to see if it works as anticipated. If there are any problems then the Coding stage is repeated to try and solve the problems.
- **Evaluation:** This is the final stage of the model and it is where the system is evaluate to see if it solves the problem that it was created for.

This methodology would be suitable for this project; this is because it splits up the development process into a number of simple tasks which when completed lead on to the next stage. Therefore it would be easy to plan out a schedule for the project.

2.2.3 Iterative Development

This methodology was created as a result of the problems that some people had with the Waterfall Model. Iterative development involves completing the system requirements and analysis as in the Waterfall Model, but then the rest is split up into small parts and so the system is gradually built up with more and more features being added. This allows for valuable feedback from the system users and in effect each iteration is a Waterfall Process in itself. This means that the system provides faster results, requires less up-front information and overall offers greater flexibility. However it is not really suitable, as the whole project specification is unlikely to change and so there is no need for the large amount of flexibility that this model provides.

2.2.4 Methodology Choice

The author decided to use the Waterfall Model as the methodology for this project. This is because it is simple and systematic, and will allow the different tasks to be broken down into manageable parts and set deadlines for when they have to be completed by.

Chapter 3

Analysis

The purpose of this chapter is to establish exactly how the application that is being developed should function and what it should do. This will involve examining existing web traffic analysis applications and identifying their key features along with the weaknesses of these applications. The results of these findings will determine the features and requirements that will hopefully be satisfied by the new application.

The Analysis stage is probably the single most important part of software development life cycle as it is intended “to establish what the users of the system want.”[10] Failure to successfully complete this stage will probably result in the development of an application that does not fully provide a solution to the initial problem. Another important factor relating to the success of systems development is that the system requirements should ideally be kept to a minimum, therefore only essential process are accounted for. This is because it is estimated that “only seven percent of the features of any given application are actually needed.”[11]

3.1 Requirements Gathering

The main part of requirements gathering stage will focus on current solutions that solve the problem of analysing web traffic in an attempt to identify exactly what they can do. It should then be possible to determine the requirements that a new application will need to fulfil.

3.1.1 Current Solutions

There are already a variety of different applications available that analyse web traffic, however the success of these applications varies and there are also areas of web analysis that are not covered. There are two different ways of analysing web traffic; the first is by installing and configuring an application on a computer that has access to the appropriate log files. The second method uses the actual html pages of the web site and by using embedded HTML tags they allow an external company to collect and analyse the web traffic statistics, and these external companies then collate and analyse the information making it available to the users that require it. However the disadvantage of using the services provided by these external companies is that they either charge a monthly fee, or they require the user to have third party advertising banners on their site. Therefore this section of the analysis will focus on the other type of analysis applications, which allow the user to configure and customise them as they wish. Another reason for focusing on these applications is that they are generally available at no cost to the user.

Three of the most popular analysis applications are AWStats, Analog and Webalizer, and as they are all free software it is possible to analyse their key features and also identify any weaknesses that they have. The first thing to note is that all three of the applications support a range of different platforms (for example Windows 2000, Windows XP, Linux, etc) and they all support a range of different log formats and web servers, such as Apache and Internet Information Services (IIS). However Webalizer supports fewest log formats, as it needs to be patched to work with the IIS format and there is no support for personalised log formats. AWStats differs to the other two applications in that as well as the access logs generated by websites, it can also analyse ftp and mail server log files.

The three applications are very similar in the way that they actually analyse the log file and display the results in different sections such as:

- Traffic data for a specific day
- Data on the number of visits from different browsers
- Data on the different keywords/key phrases used in search engines to find the page
- The most popular page/requests over a certain period
- Data on the different operating systems and technologies (Java, Flash, etc) used by visitors.

This wealth of information is displayed by all three applications in XHTML/HTML format using a variety of different charts and diagrams, etc to allow users to easily analyse trends and patterns in the data. One difference with AWStats and Analog is that they allow the user the choice of choosing whether to update the page statistics via the command line or by using the dynamic update function that uses CGI technology to automatically update the pages. The fact that the information is displayed as either HTML or XHTML files means that it can be published on the web and this is advantageous because it means that the user can view the stats from any where in the world without having to need to have a copy of the application running on the machine they are using. Although in order to reconfigure the application’s settings, the user needs to have access to the machine on which the application is running. Another advantage of using XHTML/HTML to display the information is that the way in which the data is presented can be changed easily without having to know a great deal about programming. Also by using features such as stylesheets (CSS) the various pages can be tailored to match the style of the website that is being analysed.

Along with being able to customise the appearance and layout of the analysis reports, the three applications allow the users to specify exactly what is included in the reports, so for example one user might just want to know information about the pages that are the most popular and information on their peak usage. Whereas another user may want information on the different technologies that are used/supported by visitors to the site, so the applications would allow them to specify exactly what information they require. Another common feature is that the applications provide information about the different HTTP status codes, for example they display the frequencies of ‘404 Document Not Found’ errors for example. However this is the only occasion when any of the applications comments on the errors that have occurred with a website, and so this appears to be a weakness as the error log is not used at all.

Finally there are some features that some applications support and the others do not. For instance, AWStats and Analog eliminate the data generated by ‘robots’ when they visit the websites. ‘Robots’ are generally search engine crawlers and they visit the pages of websites in order to gather information about the content, these visits will be recorded in the access log and would normally be counted as genuine visits. However they are not and so AWStats has the ability to filter out this information and provide accurate visitor statistics. Another feature of AWStats is that it allows the user to save the data from reports as either an XML file or as a structured text file. Webalizer also allows the user to save the information as a text file, however analog does not allow for this, although it is possible to use scripts written in a language such as perl to transfer the report data into a database.

3.1.2 Table comparing the current solutions

The following table compares the three applications that were looked at in 3.1.1, so that it is possible, at a glance, to see what each application can do. The information is based on AWStats comparison table. [12]

Feature	AWStats	Analog	Webalizer
Operates on a range of platforms (Linux, etc)	✓	✓	✓
Open-source/free software	✓	✓	✓
Support for a variety of log formats	✓	✓	✓
Analysis of web/ftp/mail logs	✓/✓/✓	✓/✗/✗	✓/✗/✗
Displays results in graphical form	✓	✓	✓
Range of stats relating to visits, users, etc.	✓	✓	✓
Information on visitors country/region/city	✓/✓/✓	✓/✗/✗	✓/✗/✗
Internet service provider information	✓	✗	✗
Report and filter visits by robots	✓	✓	✗
Report keywords/ phrases from search engines	✓/✓	✓/✗	✗/✓
Information on status codes (404, etc)	✓	✓	✓
Report page errors and give details	✗	✗	✗
Save analysed data	✓	✗	✓

3.1.3 Problems with current solutions

After reviewing the current solutions that are available to analyse log files it is apparent that there are problems in the way that they deal with errors that occur on a website. The problem is that the aforementioned applications only report the frequency of errors and there is no mention of exactly when they occurred and what should be done about them. This is a problem because it means that the users of the applications have to manually check the error log file, which can involve searching through many entries, in order to establish exactly what the error is and what should be done about it. Apart from that the only other potential problems would be in the set-up and configuration of the applications as it can be difficult to understand exactly what is happening, however there are detailed tutorials available on the Internet that guide the users through these processes. Once the application is correctly set-up then the user needs to do nothing else unless they change the log format or move the file to another location, but once again the resultant problems would be easily rectifiable.

3.1.4 Conclusion

After reviewing the current solutions that are available to analyse web traffic it is apparent that they are all very similar in the way that they operate, however it appears that AWStats has the most features and does pretty much everything that the user would want it to do. Therefore it would be pointless in developing another application that attempted to analyse log files in a similar way. However there is a problem with these applications in that they do not deal with the error log very well and in fact in the reports that they generate there is very little mention of them. This means that there is an opportunity to develop an application which runs separately to the actual web traffic analyser and its sole purpose is to deal with the detection and identification of errors on the website. This would save website maintainers and webmasters from having to search through the error log in order to identify errors and also would alert them when an error happens, instead of them having to come across it themselves first, which could take quite a long time.

3.2 User Requirements

The findings from the review of current solutions and the opinions of the author, who has experience of maintaining websites, were used to determine the following functional requirements of the solution. These were then classified into the minimum requirements that would be implemented into the system and the additional requirements that could be implemented if there is sufficient time available.

3.2.1 Minimum Requirements

The system shall:

- Periodically check the error log for new errors.
- Send a regular summary of errors to the user (e.g. hourly, daily, etc).
- Report when the error occurred.
- Report which page/file caused the error.
- Report the type of error that occurred.
- Report possible solutions to the error.
- Allow the user to determine when to check for errors.
- Allow the user to determine when to receive alert messages.
- Work using the command line.
- Allow users to save reports.
- Be user friendly and quick to use.

3.2.2 Extended Requirements

The system shall:

- Alert the user as soon as an error occurs – i.e. real-time error detection.
- Allow user to customise the alerts that they receive.
- Operate on a number of different operating systems.
- Provide the option to fix certain errors.
- Provide a graphical user interface (GUI).
- Scan a website for errors and report any that it finds (e.g. broken links, etc).
- Allow for integration with other software applications (e.g. databases, etc.)
- Be web-based, and so accessible to the user wherever they are.

Chapter 4

Design

4.1Initial Choices

Before any design procedures could take place, certain choices had to be made as to which platform, language and libraries would provide the basis for the application that was to be created. Once these choices were made then it would have been difficult to change them during the construction of the system, especially with regards to the programming language that was used.

The two platforms available to the author were Microsoft Windows and Linux, both displayed various advantages and disadvantages in relation to the project. For example, Windows probably the most widely used operating system with approximately eighty five percent of client side operating systems sold in 2004 were from the Microsoft Windows family [13]. Therefore due to the widespread use of Windows, this platform would offer a huge population base for products to target whilst being deemed to be user friendly to even the most novice of users.

Linux does not have a similar market share to that of Microsoft, and it is generally seen as being less user friendly to people who have never experienced using it before. However it does benefit from better system stability and performance, which would be two key concepts in creating a robust and successful product. Linux also benefits from free licensing and a whole host of tools and programs to facilitate software development, the majority of these also being freeware. However, these tools are becomingly increasingly more available on Microsoft Windows as well as Linux.

Where programming language is concerned C, C++, Java and Python were some of the main contenders due to their extensive libraries and the fact that they also support the use of regular expressions. The latter is vital, as it would be used to split up each line of the log file into different sections. These four languages can also be used on the aforementioned platforms, and so regardless of the platform they will be viable options to the author.

The C programming language was discounted due to the fact it failed to support Object Oriented development and also because the author had never worked with the language before. Object orientation is important in the development of the application because the objects that are created are portable and this allows for the reuse of code, which can save a lot of time and effort [14].

C++, Java and Python do make use of object orientation. Java provides excellent support for developers allowing them to develop code in a dedicated runtime environment. Java’s uses a simplified approach to programming that removes many complexities that can cause problems in other languages. Python is an interpreted and interactive language that uses very clear syntax and appears to be simpler than other languages, however it can still be used to create powerful applications.

The C++ language provides excellent performance and optimisation characteristics like that of its predecessor, C, although it is more complicated than the likes of Java and Python due to its diverse syntax and many extensions. However it is the language that the author has the most experience of using throughout his degree, and therefore it is the most efficient and straightforward choice for this application.

Therefore to surmise the application will run on the Microsoft Windows platform, this is because the author has access to this platform at home and so implementation can be done there, whereas with Linux the author would have to use the computers at the University of Leeds and so would be less practical. The application will be implemented using the C++ programming language.

4.2 System Design

After the functional requirements had been developed and the programming language and platform decided then it was possible to develop a program structure design that would be capable of fulfilling the requirements.

4.2.1 Technical Design

There are different ways in this application could have been structured in order to fulfil the functional requirements.

One of the possible ways would have involved developing an application that inputted the information contained within the log file into a database. Then using SQL commands and queries it would be possible to manipulate the information using a database management system such as MySQL. However this would then mean that the user would need to have a database management system installed on their computer, they would also have to know how to use it and also the application would have to cater for a range of possible database management systems, so that the widest possible user base could use the product.

The direction that was taken for this project involved producing a standalone application that consisted of just one executable and so therefore reducing the complication that is created by relying on other applications. To begin with the application imports the entries in the selected error log into an array, this is done using the stream input functions that are provided by C++ in the <*iostream*> library.

The next task that the application has to handle is the splitting up of each line of the array into the various sections. For example, as shown in Chapter 2 a log file has 6 different sections. In order to split up these sections Regular Expressions must be used. A regular expression is a string that describes or matches a set of strings, according to certain syntax rules making it possible to manipulate strings. The following example shows how a regular expression could be used to split up a line from a log file.

Log file entry:

[Wed Nov 3 10:51:34 2004] [error] [client 65.214.36.54] File does not exist: /home/tony/www/leeds-camra.com/www/General/licensinghours

Performing the following operation would remove the *[error]* and *[client* entries which are not needed and also the ‘*[*’ and ‘*]*’ characters.

re.compile(‘([error]|[|])’)

The entry would now look like:

Wed Nov 3 10:51:34 2004 65.214.36.54 File does not exist: /home/tony/www/leeds-camra.com/www/General/licensinghours

Then by removing all of the whitespace from the entry, we would have the different sections of the entry that then need to be assigned to certain variables.

re.compile(r'\W+')

Once the entries of the log file have been split up into the different sections and assigned to variables within arrays. Then it is just a matter of using simple programming techniques to manipulate the data and produce the various reports and summaries.

4.2.2 Application Processes

The application incorporated the following main processes, to allow the users to achieve their goals:

- **View Latest Error Report:** User can view all of the errors within the current 2-hour period.
- **View Daily Summary:** User can view a summary of the errors that occurred during the current day
- **View Weekly Summary:** User can view a summary of the errors that occurred during the last week
- **Search for Errors:** Search for errors depending on certain conditions (e.g. error type, IP, date)
- **Error Alerts:** The application will alert the user by email when errors occur within the last 2-hour period.
- **Save Reports:** It will also be possible for users to save reports, probably in XML format so that they can be used with other applications.

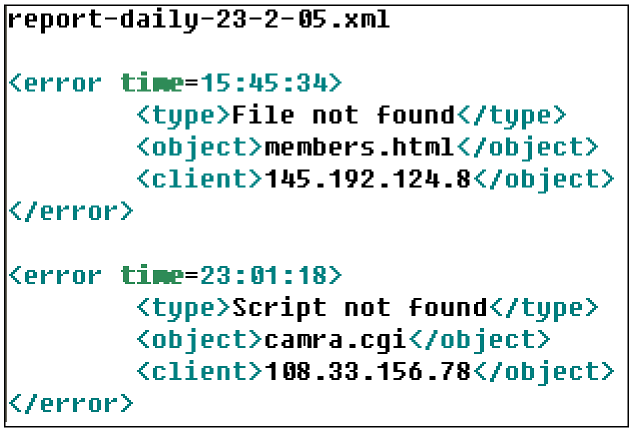
4.2.3 Workflow Diagrams

A selection of detailed workflow diagrams illustrating the main application processes can be found in Appendix C. This was used to help develop the application, ensuring that all of the processes were performed correctly and thus reduce the amount of errors that occur. It has also been used to develop the test plan for the completed application.

4.2.4 User information

The application needs access to information relating the settings that the user has configured. The application also needs to store information relating to where the log file location is and also the user has the option of saving the reports to a file.

It would be simple to implement a method of storing this information as a plaintext file with data stored as comma separated list, however the use of the structured XML format makes the files easier to read, manipulate and update. For these reasons XML files were used to store the users information and also save error reports.



```
report-daily-23-2-05.xml

<error time=15:45:34>
        <type>File not found</type>
        <object>members.html</object>
        <client>145.192.124.8</object>
</error>

<error time=23:01:18>
        <type>Script not found</type>
        <object>camra.cgi</object>
        <client>108.33.156.78</object>
</error>
```

Figure 4.1 – sample XML file

4.3 Interface Design

The user interface is a key component of any application as it is the means by which the user interacts with the application. Therefore a clear and well designed should be provided for the user, this is so that it helps the users meet their goals rather than just allowing them to accomplish their tasks [15]. If however, the system interface is badly designed and difficult to use, the repercussions negatively impact on user performance, cause frustration, affect the usage of the system and may even result in system abandonment. It does not matter what a system can do 'under the hood' if users cannot or do not want to use it due to a poorly designed interface [16].

An important requirement of the application is that it must be easy to use and allow the user to quickly view the information that they require. In order to achieve this then it was necessary to focus on goal directed design ideas taken from [15]. These ideas included focusing on the user and what goals they wanted to achieve, concentrate on functionality before presentation, keep the users tasks as simple as possible and provide feedback to the user on exactly what is happening.

As this application will be driven and displayed using the command line so the user interface will be text based and the user will enter their selections using a keyboard. Due to this the menu's and reports that the application uses will have to be clear and concise so that the user can quickly understand what is happening. By following these guidelines a design was produced with a clean appearance that does not detract from the users goals.

4.3.1 Menu Screens

Main Menu

This screen is the first one that will be seen by the user when the start the application. Therefore one of the key decisions that had to be taken was deciding what should be included on the menu. If the menu was too cluttered and contained a massive list of options then it would make the users navigation difficult, alternatively if there were very few options on this screen then the user might not be able to easily their goals. It was decided that there would be 6 options on this screen. They would be there to allow the user to view the latest error reports and summary pages, search for errors, change the applications settings and finally quit the application. To select one of these options the user simply enters the number or character that is next to the option. For example entering a '1' would view the latest error report.

It was also decided to include information about the current website log file that is selected and the time of last error report that was created. This is so that when the user starts the application they immediately know which log file is being analysed and when the last report was created.

```
errorFinder v1.0.1                      10/03/05  14:00:35
***********************************************************
                  Welcome to errorFinder
***********************************************************
Menu options:

       1:- View latest error report
       2:- View Daily summary
       3:- View Weekly summary
       4:- Search for errors
       5:- Settings

       q:- Quit

Current website: leedsCAMRA
Latest Report:   14:00:00

Please enter your selection:_
```

Figure 4.2 - Main menu screen

Settings Menu

This menu will be similar to the main menu and will use an identical selection method, where, however it will give the user of changing the various settings that affect the behaviour of the application. For example, it will allow the user to change the log file that is being analysed or change how frequently the log file is checked. It will also allow the user to select any IP addresses that they want to ignore and so these will be omitted from the search results.

4.3.2 Error Reports

The key feature of the application is the ability to view reports of errors that occur for the selected website. These reports will allow the user to view different information for each error and they will also give the user the chance to save them. The error reports will be produced every 2 hours, although there is scope in the requirements to allow the user to set the intervals at which the reports are produced. An example layout for a report can be seen below.

```
*******************************************************************
<<ERROR REPORT for leedsCAMRA from 12:00:00 to 13:59:59>>
----------------------------------------------------------

Errors Found:   3
Types of Error: 3

1 of 3:
=======
  Time:   12:04:28
  Type:   File does not exist
  Object: /home/tony/www/leeds-camra.com/www/Festival/index.html
  Client: 66.194.6.81

2 of 3:
=======
  Time:   12:38:42
  Type:   Directory index forbidden by rule
  Object: /home/tony/www/leeds-camra.com/www/Fest2003
  Client: 216.88.158.142

3 of 3:
=======
  Time:   13:01:14
  Type:   User 49283 not found
  Object: /Members/secure.shtml
  Client: 80.176.139.216
-------------------------------------------------------------------
<<END OF REPORT>>

‘m’ to go to main menu, ‘s’ to save report, ‘q’ to quit
Enter your selection:_
```

Figure 4.3 – Example 2 hourly error report

4.3.3 Error summaries

Another important feature of the application are the summary pages that can be produced, these are designed to give the user a quick overview of the errors that have occurred and also allow them at a glance to see if anything is wrong. Due to this the author tried to keep them as concise as possible and they display the total number of each type of error that occurred, which website it is that has been analysed and the time period which the statistics are from.

```
Date: Wed, 2 Mar 2005 11:36:29 GMT
From: errorFinder
To: webmaster@websiteX.com
Cc:
Subject: Error Alert – websiteX
------------------------------------------------------------------
Website: websiteX
The following errors occurred between 12:00:00 and 13:59:59

Type                           Freq.
====================================
File does not exist              7
User not found                   1
Script not found                 1
```

Figure 4.4 – Example error summary e-mail

4.3.4 System messages

One of the requirements for providing a highly user-friendly interface is that it provides good unambiguous feedback relating to the action that they have just taken. Therefore the error finding application needed to be able to alert the user to any errors that occur during the various operations, and confirmation messages for processes that have been completed successfully.

These have been provided by displaying different messages on the command line before, during and after an action has been taken.

```
Current log file location: C:\logs\error.log
Please enter new location: C:\error.log

Searching…
Error!!! Log file not found!
Try again: C:\logs\web\error.log

Searching…
Log file found
Location successfully changed.

Press enter to return to menu:_
```

Figure 4.5 Changing a log file

4.3.5 User manual

To allow users to understand the way the application works, and in line with the deliverables set out in the introduction section a user guide will be produced. The application has been designed to be user-friendly and so it might be possible to use the program without referring any documentation. However in order to ensure that the application is usable by all, from novice users through to expert users a user manual is required.

Chapter 5
Implementation

5.1 Implementation methodology

As stated in Chapter 2, the author chose the waterfall model as the methodology that would be used for the overall development of this solution. However for the implementation of the application it was decided by the author to use a methodology called evolutionary prototyping. This involved developing a high quality prototype that can be modified and built upon by the next version of the prototype.

During the implementation of this solution three different prototypes were developed and a brief overview of each of these prototypes will follow below. However before the implementation stage could begin, the software required to develop the application was installed on the author's computer by downloading the latest release of Bloodshed's C++ compiler, Dev-C++, along with the various C++ libraries that were required. The executable provided by Bloodshed made the installation simple, as the procedure was largely automatic with all the standard libraries being installed and the option to install any additional ones that might have been required. Throughout the implementation process the author consulted [14,17] in order to overcome problems that were encountered.

5.2 Prototypes

5.2.1 Evolutionary model one

The first version that was developed concentrated on the process of reading the information from the log file into the application and then splitting it up. To start with the application split up the log file line by line and it gave the user the option of printing out a selected line on the screen. This was done so that it was possible to tell whether or not the application was splitting the file correctly.

Once this process had been completed, the prototype was then modified so that it split the log file entry into the different sections. This proved to be slightly more complicated as initially there was just a one dimensional array, however as each entry was now split up into the different sections then this required the use of a two dimensional array. Once this issue had been resolved then it was possible to create a primitive menu system that allowed the user to select which element of the array they wanted to display on the screen by selecting a row number and a column number. At this point it was decided that the next prototype should be constructed as the first part of the application, the input and sorting of the log file had been completed and this code was used to from the basis of the next version.

5.2.2 Evolutionary model two

The second version that was implemented concentrated on the menu system that was to be used to allow the user to choose what they wanted to do. This was done using a switch statement whereby the user selects a specific case and depending on their selection the application performs a defined task (for example, display the latest error report). This was relatively straightforward to achieve and after completion it was possible to focus on the functions that would create the various reports.

The application was designed to be able to produce reports every two hours and in order to this a function had to be developed that would take the check the time when the last report was created, calculate the time the next report was due and once this time arrived then the application would check the log file using the methods from the first prototype to establish whether any errors had occurred and report them to the user.

This was probably the most challenging aspect of the implementation as the exact methods for creating the reports had to be finalised and various algorithms for establishing when the reports should be created had to be developed. Although once they had been created it was rewarding moment as the application was then virtually complete and it could take in the entries from the log file and create reports with them based upon the users actions.

5.2.3 Evolutionary model three

This final version of the application involved adding all the finishing touches, these included finalising the menu’s that were used, adding the time and current log file that was being analysed to the main menu. This version also dealt with the different settings that the user could manipulate, including how often checks are performed, how the reports are sorted (e.g. by IP, by error type, etc), any IP addresses they wished to ignore, and also it gave the user the option to change the location of the log file.

5.3 Problems during development

During the development of the application a number of problems occurred. Apart from the ones that have been mentioned previously, one problem was that the log file was being overwritten every time a check was made. As a result of this it was not possible to the user to view information about errors that had occurred earlier in the day. This problem was rectified by creating a separate copy of the log file that was used by the application to check for errors. So when this file was overwritten it did not matter, as the original file was still intact.

There were also problems with the user interface and they were mainly design and layout one as opposed to functional problems. The root of these interface problems was that the author failed to take into account the fact that different users will have different screen resolutions and different window sizes, etc. Due to this the layout did not work on smaller terminal window resolutions, an instance of this was that the menu did not align correctly when it was viewed in a small terminal window as opposed to a maximised window. In order to overcome this problem the author limited the number of characters per line that were to be displayed on the screen, this meant that even on a small window the menu’s were aligned and generally moor aesthetically pleasing than previously.

Chapter 6

Testing

The fact that evolutionary prototyping was used to create the application meant that testing was performed on a continuous basis during development to remove bugs that were detected in each component. However despite this it was decided that a more rigorous test procedure designed to fully test the all of the system components together would be beneficial, this is because it would highlight any problems that the user may encounter.

In order to test the application fully it was split up into two sections. The first was Unit testing and the second was System testing. This approach was identified by [18] in order to ensure that the system would function as expected for the vast majority of inputs received, and paths of program execution.

6.1 Unit Testing

For the unit tests the separate components of the system were tested for functionality, separate from each other. The tests that were used for the unit testing were devised from the source code of the application in an attempt to stretch the application and identify any flaws. The testing was of an ad-hoc manner, this was due to the unpredictable nature of developing code and so when an error occurred a solution was implemented immediately.

6.2 System Testing

In order to thoroughly test the final application, a comprehensive test plan was developed that would test the system as a whole without using specific knowledge of the underlying code, but by performing the operational tasks that an application user might wish to achieve. This was done using a logical walk through in an attempt to cover as many possible paths through the system as possible. A Test plan complete with results for these tests can be seen in Appendix D. Very few further errors were discovered within the system testing cycle, justifying the evolutionary development of the application and the manner in which the unit tests were carried out.

6.3 Testing Summary

Whilst the testing of the application has been successful and the author has rectified all of the problems that they have found, it is not possible to completely test an application. This is because each different user will use the application in a different manner and so there are different errors that may occur when they use it as opposed to when someone else uses the system. Therefore fully testing the application for all possible inputs and paths of execution would take an immense amount of time. The author is confident that all of the less obscure and most likely problems have been solved producing a reasonably stable application.

Chapter 7
Evaluation

7.1 Product Evaluation

This aim of this section is to decide on whether the application produced as a deliverable for this project meets the specification that were originally set out for it. [19, 20] have been consulted in order to verify the solution against the requirements set out for it during the analysis stage, while usability evaluation has been carried out to evaluate the solutions effectiveness. Comparisons have also been made to existing solutions identified earlier, in order to evaluate the success of the application in solving the original aim of the project.

7.1.1 Product Requirements Review

The original product requirements developed during the analysis stage of the systems life cycle are shown below, a tick marks the requirements that have achieved.

Minimum requirements

The system shall:

- ✓ Periodically check the error log for new errors.
- ✓ Send a regular summary of errors to the user (e.g. hourly, daily, etc).
- ✓ Report when the error occurred.
- ✓ Report which page/file caused the error.
- ✓ Report the type of error that occurred.
- ✗ Report possible solutions to the error.
- ✓ Allow the user to determine when to check for errors.
- ✗ Allow the user to determine when to receive alert messages.
- ✓ Work using the command line.
- ✓ Allow users to save reports.
- ✓ Be user friendly and quick to use.

Extended Requirements

The system shall:

- ✗ Alert the user as soon as an error occurs – i.e. real-time error detection.
- ✗ Allow user to customise the alerts that they receive.

- ✓ Operate on a number of different operating systems.
- ✗ Provide the option to fix certain errors.
- ✗ Provide a graphical user interface (GUI).
- ✗ Scan a website for errors and report any that it finds (e.g. broken links, etc).
- ✗ Allow for integration with other software applications (e.g. databases, etc.)
- ✗ Be web-based, and so accessible to the user wherever they are.

Of the minimum product requirements, the majority of requirements have been implemented with the exception of the one relating to reporting of possible solutions to errors, and the one that allows the user to choose when they receive alerts. These were removed from the requirements during the implementation stage due to technical difficulties relating to the programming language and time limitations.

In terms of extended requirements it was only possible to implement the one relating to the portability of the application, in that it actually worked on both the Windows and Linux operating systems. In reflection, the requirements developed for the application might have been a little unrealistic in specifying a range of functions, when the product may have proved to have as much value to users without some of these features.

7.1.2 Design Evaluation

The completed product closely resembles the original design ideals. The application itself runs from the command line as originally intended, and in the author’s opinion the application is reasonably simple to use without much prior knowledge of computing. The implementation method that was used produced a program with a structure that very closely resembles the workflow diagrams (Appendix C). The user interface is also very similar in terms of look and feel to those that were produced in Chapter 4.

7.1.3 Usability Evaluation

A key aim of this project was that the application that was developed should be simple to use, therefore user-friendly for even the most inexperience users, whilst still managing to make it easier for website owners and maintainers to get the most out of the information that there log files generate. To determine whether this was the case a usability evaluation was carried out on the competed product with a sample of website maintainers that the author knows. The evaluators consisted of Jamie Kennaway, a student at the University of Leeds and the developer of purpleplacements.com, Neil Thompson, the administrator and owner of wanderersways.com, and finally Nick Wilkinson, the owner of takeda-telecom.co.uk. To allow the evaluators to get to grasps with the systems a set of exercises (Appendix E) were developed for them which took them through the key features of the system.

The evaluators were provided with access to the application and a set of sample log files to edit. They were also given a copy of the user manual, and were asked to perform the tasks under observation by the author. All of the evaluators managed to complete the exercises successfully with little or no help from the observer. All of the participants initially consulted the manual to get an idea of how to run the application, but after that the all of them found the tasks reasonably straightforward. All of the evaluators stated that despite being primitive they could see the ways in which the application could assist with the task of analysing the error logs produced by websites.

Of the three evaluators Jamie Kennaway was the only one who currently used a web traffic analyser and expressed approval at the way it displays the information about the errors, a feature that is missing from the analyser he uses. Also being a graphic designer, he indicated that he would like to see a GUI version that could display information in charts and graphs, therefore making it even easier to take in the information.

Overall the evaluators were pleased with the application would like to see it further developed and the features linked in with a pure web traffic analyser such as Analog.

7.1.4 Comparison with Existing Systems

The application produced does not suffer from any of the problems that exist with the current solutions discussed in earlier. This is expected because the aim of the application was to solve some of the problems that occur with these existing applications. However the noticeable difference when compared to these existing solutions was that it looked less professional than them and in fact less user friendly. This is because they use HTML pages to display the information in a web browser, which means that results are displayed as graphs and the browser technology that the user has to interact with is already familiar to them. Whereas programs that run from the command line can be daunting to inexperienced users.

7.1.5 Product Evaluation Summary

The application provides a way for all levels of user to obtain the information that is contained in the error log files produced by websites. The system has a user interface and operational structure that meets the original desire for an application that is user-friendly and quick to use. Although when compared to the existing solutions that have been developed over a much longer period of time, and by professional developers the product does seem to be lacking the ‘wow’ factor.

The application produced solves the problem posed at the start of the project by improving on the weaknesses of existing solutions, and therefore meeting the requirements that were specified.

7.2 Project Evaluation

This section evaluates the project based on whether the original objectives and minimum requirements set out in Chapter 1 were achieved.

7.2.1 Project Objectives and Requirements Review

All of the objectives set out at the start of this project have been achieved resulting in a full set of deliverables. It also proved possible to complete the minimum requirements developed from the objectives. However when the feedback from the mid-project report was issued the direction of the project had to change. This is because initially the project was going to solve the problem of analysing the access log, however after the completion of the analysis stage and consultation with the assessor of this project it was decided that the project should concentrate on the weaknesses of existing solutions, i.e. the inability to properly analyse the error log file. This means that the extended requirements were no longer valid.

7.2.2 Development Methodology Evaluation

The development methodology used proved to be well suited to a project of this nature, because by separating the project into a number of distinct stages, or milestones, it provided a stable structure to the entire project. The waterfall model also provided a lot of flexibility in that it can differ depending on the project type, so this lack of rigidity meant that changes to schedules and plans could be made. Also in the implementation section it allowed for a change of methodology due to the nature of the work to be done.

The separate stages of the methodology also meant that it was easy to identify when the project was not going according to the schedule, allowing for changes to the rest of the stages in order to compensate for any delays. Also at the beginning of semester the methodology allowed for re-evaluation of the analysis section in line with the recommendations made by the assessor. With a different methodology this delay might have proved even more costly, to the point of where the project may have failed to be completed.

7.2.3 Development Language Evaluation

In hindsight C++ may not have been the best language to create this application in. This is because it became apparent that it would have been easier and much quicker to complete various sections if another language had been used. The reason why C++ was used was because of its familiarity to the author, with the idea that there would be less learning involved. However it turned out that time was needed to look up various things in order to proceed to the next stage, and in fact the authors knowledge of C++ barely scratched the surface of the complex programming language.

7.2.4 Project Evaluation Summary

Despite the fact that problems were encountered with the chosen language, and also the changes that needed to be made to the early stages of this project, most of these were quickly rectified while any delays were identified and resolved, thus ensuring the project was completed on time. Most importantly the product produced by this project solves the problem that was set out at the beginning, as shown by the requirements verification and as it was a learning process there were things that would be done differently if it were to be repeated.

Chapter 8

Further enhancements and Conclusion

8.1 Further Enhancements

A number of enhancements to this product still remain from the original extension requirements, while others have been identified during the course of the systems development life cycle.

Of the unimplemented extended requirements, it is felt that the most important ones are the ability to alert the user to errors that occur when they occurs, and that the system should be extended to allow the use of a GUI that would vastly improve the users experience. The other requirements are less important, as for example providing the ability to solve errors would require a vast knowledge base that would cover every conceivable problem. Also users like to know exactly what is happening to their website and may not be keen on automatic processes that alter the site.

Other possible extensions would benefit the application would be the ability to link the application with existing applications that analyse the remaining areas of web traffic analysis. This would create an all in one application that would cover all conceivable aspects and would ensure that users could obtain all of the information that they require. Also it would be an improvement if the application could be linked to a database. This would increase the scope for interaction with various other business applications.

8.2 Conclusion

Following an investigation into the user requirements for a system that analyses web traffic, an application has been produced that both focuses on the weaknesses of existing solutions and provides functionality beyond the minimum requirements identified, and has been approved as a valid solution to the original problem by the target user group.

The system has been successfully tested, evaluated and a number of possible enhancements identified.

References

[1] Internet World Stats, ***"Internet Usage Statistics"***, (24th March 2005)
Available: http://www.internetworldstats.com/stats.htm/
(Accessed: 30th March 2005)

[2] ISC, ***"Internet Domain Survey"***, (25th January 2005)
Available: http://www.isc.org/index.pl?/ops/ds/
(Accessed: 30th March 2005)

[3] Leeds CAMRA, ***"Leeds CAMpaign for Real Ale (CAMRA)"***, (1st December 2004),
Available: http://www.leeds-camra.com/
(Accessed: 7th Dec 2004)

[4] Sun Microsystems Inc, ***"Administrator's Guide: Understanding Log Files"***, (2002),
Available: http://docs.sun.com/source/816-5666-10/esmonsvr.htm/
(Accessed: 20th Nov 2004)

[5] Apache Organisation, ***"Log Files – Apache HTTP Server"***,
Available: http://httpd.apache.org/docs/logs.html/
(Accessed: 7th Dec 2004)

[6] World Wide Web Consortium, ***"Logging in W3C httpd"***, (July 1995),
Available: http://www.w3.org/Daemon/User/Config/Logging.html/
(Accessed: 20th Nov 2004)

[7] Fielding et al, ***"RFC 2616, Hypertext Transfer Protocol -- HTTP/1.1"***, (June 1999),
Available: http://www.ietf.org/rfc/rfc2616.txt/
(Accessed: 7th Dec 2004)

[8] Apache HTTP Server, ***"Apache Core Features – LogLevel Directive"***,
Available: http://httpd.apache.org/docs/mod/core.html#loglevel/
(Accessed: 7th Dec 2004)

[9] Avison D.E and Fitzgerald G, ***"Information systems development : methodologies, techniques and tools"***, (3rd Edition - 2003), McGraw-Hill, London.

[10] Bell, D, Morrey, I & Pugh, J, ***"Software Engineering, A Programming Approach"***, (2nd edition – 1992)**,** Prentice Hall, Hertfordshire, England.

[11] CIO.com, ***"The Secret to Software Success"***, (1st July 2001),
Available: http://www.cio.com/archive/070101/secret.html
(Accessed: 6th March 2005)

[12] AWStats, ***"Log Analyzers Comparisons",*** (1st July 2001),
Available: http://awstats.sourceforge.net/docs/awstats_compare.html
(Accessed: 6th March 2005)

[13] CNET News, *"**Linux closing in on Microsoft market share, study says",*** (24th July 2000),
Available: http://news.com.com/2100-1001-243527.html?legacy=cnet
(Accessed: 6th March 2005)

[14] Jenkins T, ***"How to program using C++"***, (1st Edition - 2003), Palgrave MacMillan

[15] Cooper A, "***About Face, The Essentials of User InterfaceDesign"***, (1st Edition – 1995), IDG, Foster City, CA, USA.

[16] Wickens C.D, Gordon S.E, Liu Y, ***"An introduction to human factors engineering "***,
(1st Edition - 1998), Longman, New York, USA.

[17] Deitel H.M, Deitel P.J, ***"C++ How to program***", (4th Edition - 2003), Prentice Hall, New Jersey, USA.

[18] Petrenko A, Ulrich A, ***"Formal Approaches to Software Testing"***, (2004) Spinger, Berlin, Germany.

[19] Watts R, ***"Measuring Software Quality"***, (1987), NCC Publications, Manchester, England.

[20] Lewis Robert, "***Independent Verification and Validation, A Life Cycle Engineering Process for Quality Software"***, (1992) John Wiley & sons, USA.

Appendix A

Personal Reflection

I feel that this project went well, especially because of the fact that everything was completed on time and before the deadlines. This is despite the fact that I had to change the direction of the project at the beginning of semester 2. With respect to the application I feel proud that it succeeded in achieving the requirements that were developed for it and real world users were also pleased with the features that it offered.

I feel that this project has given me a great deal of experience in managing a project from its inception to its completion; I hope that these experiences will serve me well in my chosen career path. I also think that my communication skills have been improved in terms of actually liasing with different people regarding the project and also from having to write up the project report, which is something that I never thought I would be able to manage.

I have learnt a number of valuable lessons during the completion of this project, and after discussions with other students who have also completed projects; I feel these lessons should be shared so that they would benefit other students in a similar situation. They are described below.

Choose an interesting project: This is one of the key motivators for the project, because if you are not interested in the project then you will not be as keen to complete the various tasks. Therefore after 6 months, you may find that motivation is lacking due to a loss of interest and so you will struggle to complete the project. Think carefully about exactly what project you want to do and take the time to ensure that it is the correct one for you.

Make appointments as early as possible: During the project you will have to meet various people, who will only be available at certain times. Therefore don’t rely on everyone being free when you are. This issue proved a problem for me because I left it until the week before to try and arrange my progress meeting and as my assessor was only in university twice a week I had to arrange my meeting for during the Easter break. This was purely my fault and in future I will learn from this experience.

Don’t be afraid to ask questions: Your project supervisor and assessor are there to help you, so use them to check any ideas that you have or ask them for any advice that you have. No matter how small these problems are or how silly the questions may be, they will always provide you with valuable advice that will benefit your project.

Don’t neglect other modules: Just because this project is worth a large percentage of your final year doesn’t mean that it should get priority over other pieces of coursework. These other modules are just as important to the successful completion of your degree. Try and plan around these modules so that you have time set aside for them whilst still being able to make progress with your project. Also at Christmas give yourself a break from the project so that you can fully concentrate on the January examination period.

Start the project write up as early as you can: The write up of the project report can seem like a very daunting task as 50 pages seems like a lot when you haven’t started. I would advise writing a draft report of each section immediately after you have completed it. This is because the information will be fresh in your mind and also at the end of the project these sections will just need to be compiled together and read through to ensure that any necessary changes are identified. Once you start the write up then it will come naturally to you and the report will be completed sooner than you think.

Appendix B – Project Schedule

Initial Schedule:

Semester 1

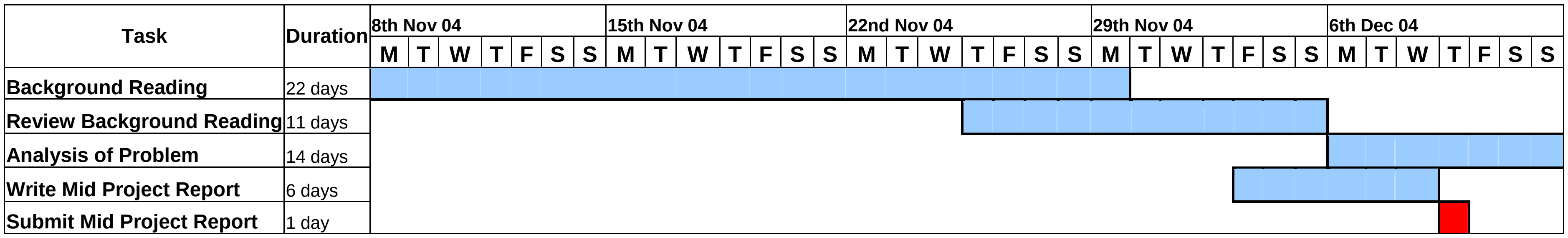
Task	Duration	8th Nov 04							15th Nov 04							22nd Nov 04							29th Nov 04							6th Dec 04						
		M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S	M	T	W	T	F	S	S
Background Reading	22 days	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦													
Review Background Reading	14 days															🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦							
Analysis of Problem	14 days																													🟦	🟦	🟦	🟦	🟦	🟦	🟦
Write Mid Project Report	10 days																						🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦				
Submit Mid Project Report	1 day																																🟥			

Semester 2

Task	Duration	Jan 05			Feb 05									Mar 05										Apr 05									
		24	27	30	02	05	08	11	14	17	20	23	28	03	06	09	12	15	18	21	24	27	30	02	05	08	11	14	17	20	23	26	29
Application Design	18 days	🟦	🟦	🟦	🟦	🟦	🟦																										
Application Implementation	26 days							🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦																	
Application Testing	12 days															🟦	🟦	🟦	🟦	🟦													
Submit TOC & Draft Chapter	1 day																🟥																
Progress Meeting	1 day																		🟥														
Application Evaluation	14 days																				🟦	🟦	🟦	🟦	🟦	🟦							
Write Up Project Report	18 days																							🟦	🟦	🟦	🟦	🟦	🟦				
Check and correct Report	7 days																													🟦	🟦	🟦	
Submit Reports	1 day																															🟥	

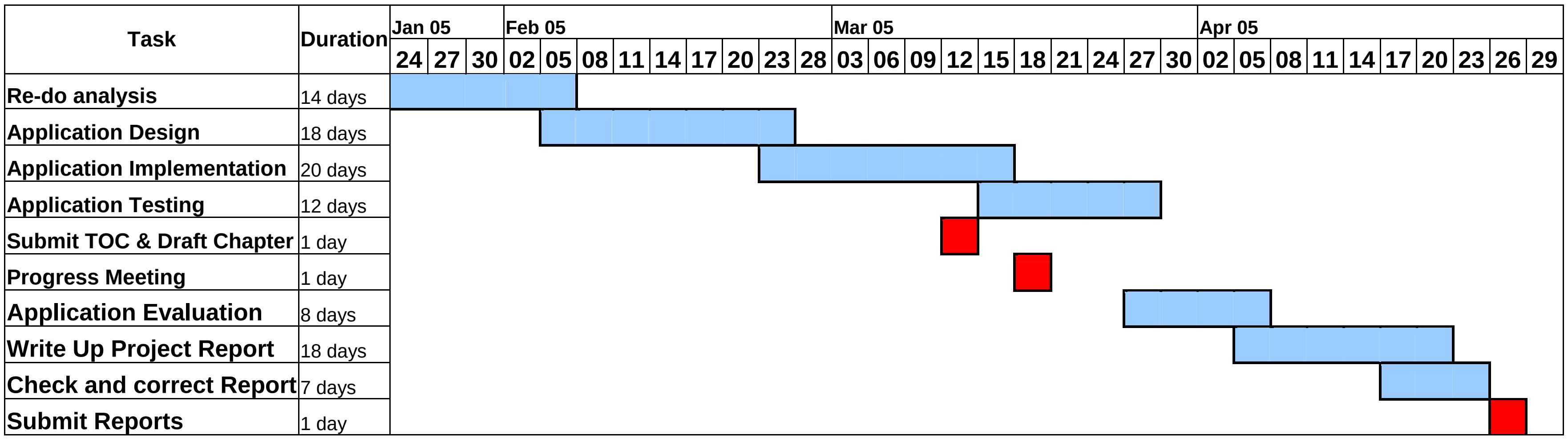
Actual Schedule:

Semester 1



Task	Duration
Background Reading	22 days
Review Background Reading	11 days
Analysis of Problem	14 days
Write Mid Project Report	6 days
Submit Mid Project Report	1 day

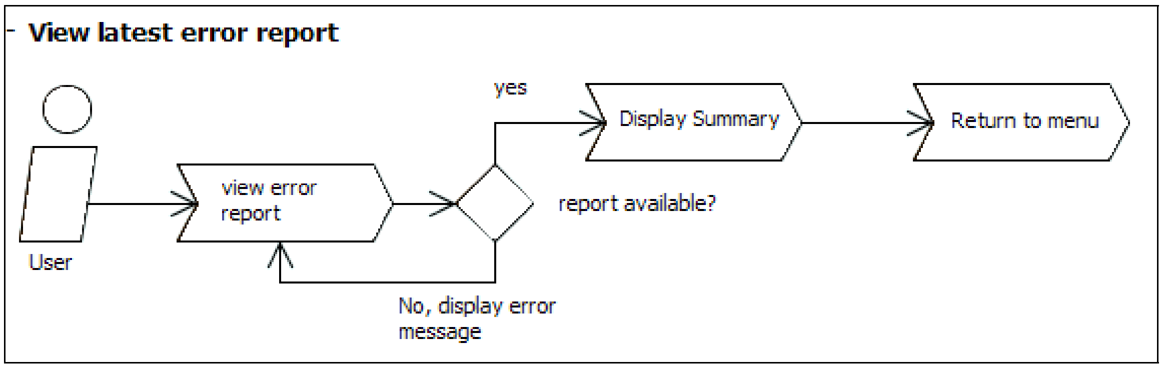
Semester 2

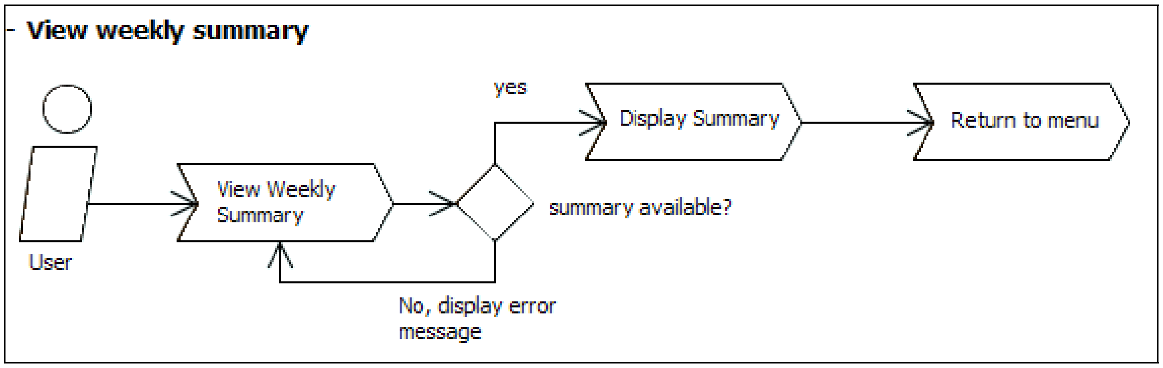


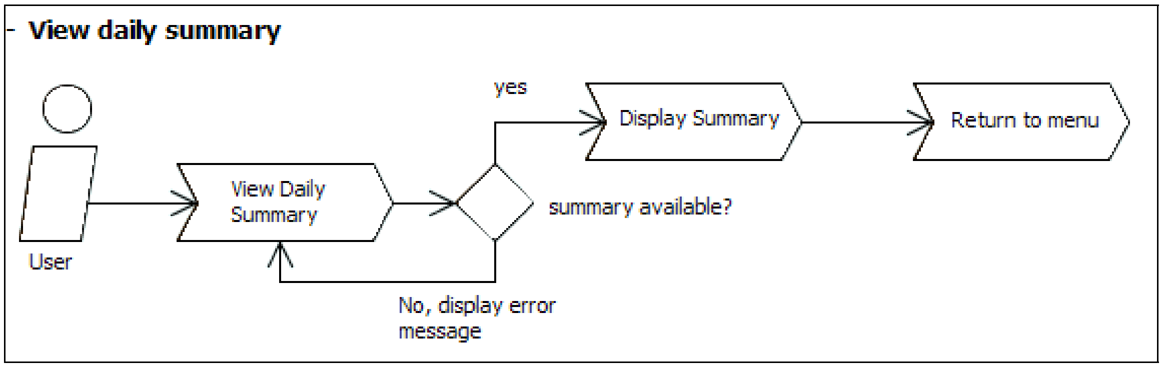
Task	Duration
Re-do analysis	14 days
Application Design	18 days
Application Implementation	20 days
Application Testing	12 days
Submit TOC & Draft Chapter	1 day
Progress Meeting	1 day
Application Evaluation	8 days
Write Up Project Report	18 days
Check and correct Report	7 days
Submit Reports	1 day

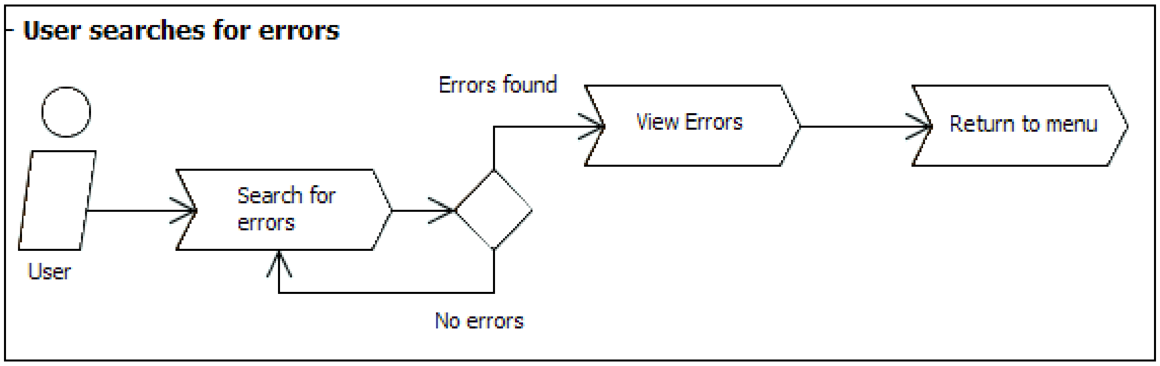
Appendix C

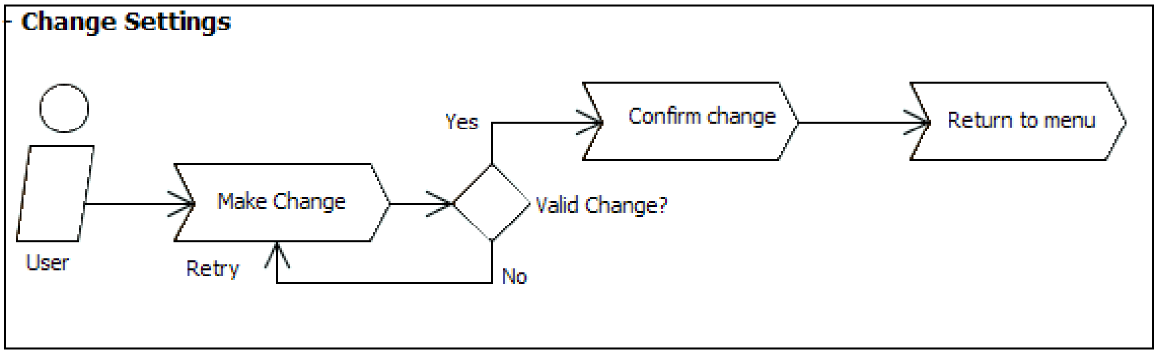
Workflow Diagrams

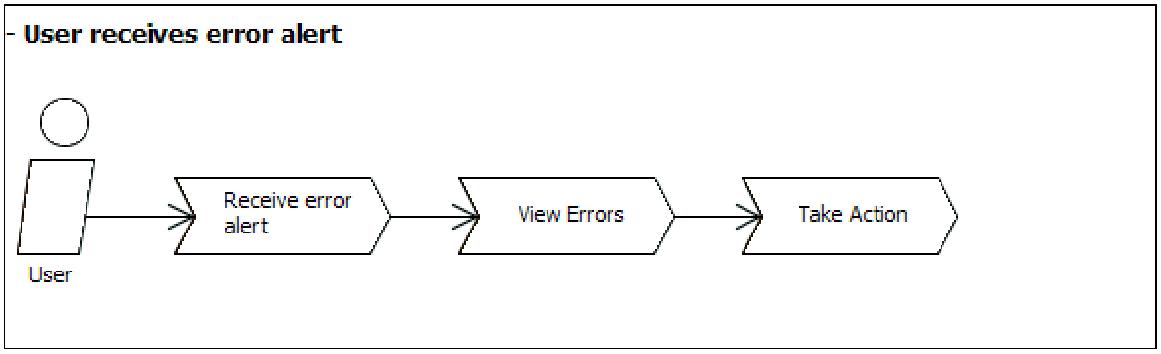







User searches for errors
User
Search for errors
Errors found
View Errors
Return to menu
No errors


Change Settings
User
Make Change
Retry
Yes
Valid Change?
No
Confirm change
Return to menu


User receives error alert
User
Receive error alert
View Errors
Take Action

Appendix D

Application Testing Plan

Function	Input/Output event	Valid Values	Value Entered	Outcome	Expected?	Pass
Main Menu	Enter selection	1 to 5, q, Q	1	Displays latest error report	Yes	✓
			2	Displays daily summary	Yes	✓
			3	Displays weekly summary	Yes	✓
			4	Displays search menu	Yes	✓
			5	Displays settings menu	Yes	✓
			q	Quits program	Yes	✓
			Q	Quits program	Yes	✓
			x	Invalid selection, try again:	Yes	✓
			8	Invalid selection, try again:	Yes	✓
			dfsdf	Invalid selection, try again:	Yes	✓
			3rf	Invalid selection, try again:	Yes	✓
Error reports/ summary pages	Enter selection	s, m, q	s	Saves to file	Yes	✓
			S	Invalid selection, try again:	Yes	✓
			m	Returns to main menu	Yes	✓
			M	Invalid selection, try again:	Yes	✓
			q	Quits program	Yes	✓
			Q	Invalid selection, try again:	Yes	✓
			3	Invalid selection, try again:	Yes	✓
			xcvx	Invalid selection, try again:	Yes	✓
			sds43	Invalid selection, try again:	Yes	✓

Function	Input/Output event	Valid Values	Value Entered	Outcome	Expected?	Pass
Search for errors	Enter selection	1 to 4, m, q	1	Search by type	Yes	✓
			2	Search by client	Yes	✓
			3	Search by date	Yes	✓
			4	Search by file	Yes	✓
			m	Returns to main menu	Yes	✓
			q	Quits program	Yes	✓
Search by type	Enter error type	Any string	File not found	Valid option, searches for errors	Yes	✓
			User not found	Valid option, searches for errors	Yes	✓
			deadlink	Invalid option	Yes	✓
Search by client	Enter client IP	Valid IP address	23.123.3.132	Valid option, searches for errors	Yes	✓
			ff.ff.33.322	Invalid option	Yes	✓
			999.54.54.34	Invalid option	Yes	✓
Search by date	Enter date	dd:mm:yyyy	21:04:2005	Valid option, searches for errors	Yes	✓
			april 5 2005	Invalid option	Yes	✓
Search by file	Enter filename	Any string	index.html	Valid option, searches for errors	Yes	✓
			home	Valid option, searches for errors	Yes	✓

Function	Input/Output event	Valid Values	Value Entered	Outcome	Expected?	Pas
Settings Menu	Enter selection	1 to 4, m, q	1	Change log file location	Yes	✓
			2	Change frequency of check	Yes	✓
			3	Change save file directory	Yes	✓
			4	Set email address for alerts	Yes	✓
			m	Return to main menu	Yes	✓
			q	Quit program	Yes	✓
Change Log file location / save directory	Enter new location / directory	Valid location	home	Invalid option	Yes	✓
			54543	Invalid option	Yes	✓
			C:\files	Valid option, makes change	Yes	✓
Change frequency of check	Enter new frequency (hours)	any integer	2	Valid option, makes change	Yes	✓
			e	Invalid option	Yes	✓
			rf34	Invalid option	Yes	✓
Set email address	Enter address	email address	jhs2rf@leeds.ac.uk	Valid option, makes change	Yes	✓
			email	Invalid option	Yes	✓

Appendix E

Usability Evaluation Exercise

1. Run the errorFinder application using command line arguments.

2. View the latest error report that is available and then return back to the main menu.

3. Go to the settings menu and change the current log file location to C:\errorFinder\logs.

4. Whilst on the settings menu, change the frequency of error checks from 2 hours to 4 hours.

5. Return back to the main menu, and then choose to search for errors for the client with the IP address of 64.78.102.1. Are there any errors?

6. Search for any errors that have occurred with the file index.html.

7. View the latest daily and weekly summaries.

8. Change the location where the application saves the reports from the default path to C:\documents\savedReports.

9. View either a summary or report and save a copy of it using the ‘s’ save function.

10. Perform any other operations that you wish to until you are happy with how the application works, then quit the application using the ‘q’ command.